



*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

## APPARATUS AND METHOD FOR WEB-BASED TOOL MANAGEMENT

## TECHNICAL FIELD

The present invention relates in general to data processing systems, and in particular, to data processing systems for managing manufacturing equipment via the web.

## 5 BACKGROUND INFORMATION

Modern manufacturing facilities rely on highly automated tools to implement the manufacturing process. For example, semiconductor fabrication ("fab") facilities incorporate highly automated tool sets for the processing of semiconductor wafers. Process control and monitoring is mediated through a set of software methods which may be invoked to implement the processes and monitoring to be performed. The control and monitoring software run on a tool server which may be coupled to the tools via a plurality of ports, each of which interfaces the tool server with a particular tool, in point-to-point fashion. Alternatively, the tools in the tool server may reside on a Local Area Network (LAN). To control the manufacturing process, a user must be able to communicate with the tool server, either via a user system resident on the LAN, or otherwise in communication with the tool server. In particular, remote access to the tool server for control and monitoring of the status of a tool, to the extent that it exists at all, requires the development of specialized code implemented on each platform for which remote access is to be provided. Thus, there is a need in the art for methods and apparatus for providing remote tool management capabilities. In particular, there is a need in the art for apparatus and methods for providing remote tool management without the necessity for developing specialized, platform dependent software.

## SUMMARY OF THE INVENTION

The aforementioned needs are addressed by the present invention. Accordingly, there is provided, in a first form, a method of tool management. The method includes receiving a request over a network. A request type is determined from a predetermined portion of the request. In response to the request and the type of the request, a message is sent to the tool, which message is operable for controlling an operation of the tool.

There is also provided, in a second form, a data processing system. The system includes circuitry operable for receiving a request via a network. The system also includes circuitry operable for determining a type of the request using a predetermined field in a portion of the request, and circuitry operable for sending a message to a tool in response to the first request and the type of the request. The message is operable for controlling an action of the tool.

Additionally, there is provided, in a third form, a program product embodied in a tangible storage medium. The program contains programming for tool management and includes instructions for receiving a request over a network. A request type is determined from a predetermined portion of the request. Also included are instructions for sending a message to the tool in response to the request and the type of the request, which message is operable for controlling an operation of the tool.

The foregoing has outlined rather broadly the features and technical advantages of the present invention in order that the detailed description of the invention that follows may be better understood. Additional features and advantages of the invention will be described hereinafter which form the subject of the claims of the invention

5

#### BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, in which:

10

FIGURE 1 illustrates, in block diagram form, a data processing network in accordance with an embodiment of the present invention;

FIGURE 2 illustrates, in block diagram form, a data processing system which may be used in a client and a server of FIGURE 1;

15

FIGURE 3 illustrates, in flow chart form, a hypertext transfer protocol (HTTP) server methodology in accordance with an embodiment of the present invention.

FIGURE 4 illustrates, in flow chart form, a security server methodology which may be used with the HTTP server process of FIGURE 3;

FIGURE 5 illustrates, in flow chart form, a process for generating responses to HTTP requests received in accordance with the HTTP methodology of FIGURE 3;

20

FIGURE 5.1 illustrates a graphical user interface (GUI) representation of a tool object model which may be used in the HTTP request process of FIGURE 5;

FIGURE 5.2 illustrates, in flow chart form, an exemplary process control methodology which may be used with the present invention;

25

FIGURES 6.1 and 6.2 illustrate displays associated with an information request in accordance with the principles of the present invention;

FIGURES 7.1 and 7.2 illustrate displays associated with an edit request in accordance with the principles of the present invention;

FIGURES 8.1 and 8.2 illustrate displays associated with an expand request in accordance with the principles of the present invention;

30

FIGURES 9.1 and 9.2 illustrate displays associated with an execute request in accordance with the principles of the present invention;

FIGURES 10.1 and 10.2 illustrate displays associated with an information request for a method history in accordance with the principles of the present invention; and

FIGURE 11 illustrates, in flow chart form, a service method override process in accordance with an embodiment of the present invention

35

#### DETAILED DESCRIPTION

In the following description, numerous specific details are set forth such as specific word or byte lengths, etc. to provide a thorough understanding of the present invention. However, it will be obvious to those skilled in the art that the present invention may be practiced without such specific details. In other instances, well-known circuits have been shown in block diagram form in order not to obscure the present invention.

40

obscure the present invention in unnecessary detail. For the most part, details concerning timing considerations and the like have been omitted in as much as such details are not necessary to obtain a complete understanding of the present invention and are within the skills of persons of ordinary skill in the relevant art.

5           Refer now to the drawings wherein depicted elements are not necessarily shown to scale and wherein like or similar elements are designated by the same reference numeral through the several views.

          The present invention provides a mechanism for managing automated manufacturing tools and processes via an Internet worked data processing system. Remote interactions between a client and a processing tool are provided using standard Internet protocols and facilities. These may include  
10           the hypertext transfer protocol (HTTP), the hypertext markup language (HTML), and Extensible Markup Language (XML). The present invention enables users to interact with an automated tool using a standard web browser running on a client data processing system. The user may, for example, query and change the status and configuration of the automated tool, and command it to  
15           perform activities, such as in a semi-conductor fabrication environment, for example, to begin processing material. The present invention also allows users to interact with a station controller interfaced to the tool, querying and changing the configuration and status of the station controller, and instructing the controller to perform various activities, for example, performing a logging or reporting function. Additionally, the methods and apparatus of the present invention include web server  
20           functionality, obviating the need to configure and administer a separate web server.

          Refer now to the drawings, wherein depicted elements are not necessarily shown to scale and wherein like or similar elements are designated by the same reference numeral through the several views

          FIGURE 1 illustrates a representative network environment which may be used with the  
25           present invention. A plurality of clients 112-115 facilitate user interactions with a plurality of automated manufacturing tools 111. Within clients 112-115, the tool management facilities are provided via a standard web browser. Such web browsers are known in the data processing art, and include, for example, Netscape Navigator®, a product of Netscape Communications Corporation, and Internet Explorer, a product of Microsoft Corporation. Interactions between users, one of clients 112-115 and  
30           tools 111 are mediated via server 110 which implements the methodologies of the present invention, further described below in conjunction with FIGURES 3-6

          Clients 112-115 are networked with server 110. Clients 112 and 113 are networked via LAN 122. Clients 114 and 115 are networked via the Internet. Clients 114 and 115 communicate via a facility having a presence on the Internet, here generically referred to as an Internet Service Provider  
35           (ISP) 118. Additionally, server 110 interfaces with the Internet via an ISP 118. Note that clients need not rely on a wired connection but may be a wireless device. Wireless clients could, for example, include a conventional personal computer and wireless modem or even a web capable cellular telephone. Each ISP 118 is coupled to Internet backbone 120. Alternatively, a manufacturing facility may itself have a presence on the Internet, and in such an alternative embodiment of the networking  
40           environment in FIGURE 1, server 110 may incorporate hardware, software, and/or firmware to

backbone 112. Information may be exchanged between clients 112-115 and server 112 using the hypertext transfer protocol (HTTP). Additionally, the data constituting the information for managing tools 111 to be exchanged between clients 112-115 and server 110 may be communicated using the transmission control protocol TCP/Internet Protocol (TCP/IP). TCP/IP is a standard communication protocol on the Internet.

Additionally, server 110 is networked with tools 111 and provides tool management information thereto in response to the user input received via clients 112-115. In the exemplary embodiment of FIGURE 1, server 110 is connected to tools 111 via LAN 124. Alternatively, tools 111 may reside on LAN 122. However, in such an alternative embodiment, tool management operations by users on clients 112 and 113 would still be mediated via server 110 through the standard web browser on clients 112 and 113.

FIGURE 2 illustrates a block diagram of a representative data processing system 200. Data processing system 200 may be used to implement clients 112-115. Additionally, data processing system 200 may be used in an implementation of server 110 by incorporating software which performs the methodologies of the present invention as described further below. The system has a central processing unit (CPU) 210, which is coupled to various other components by system bus 212. Read-only memory ("ROM") 216 is coupled to the system bus 212 and includes a basic input/output system ("BIOS") that controls certain basic functions of the data processing system 200. Random access memory ("RAM") 214, I/O adapter 218, and communications adapter 234 are also coupled to the system bus 212. I/O adapter 218 may be a small computer system interface ("SCSI") adapter that communicates with a disk storage device 220. Communications adapter 234 interconnects bus 212 with an outside network enabling the data processing system to communicate with other such systems. Input/Output devices are also connected to system bus 212 via user interface adapter 222 and display adapter 236. Keyboard 224, track ball 232, mouse 226 and speaker 228 are all interconnected to bus 212 via user interface adapter 222. Display monitor 238 is connected to system bus 212 by display adapter 236. In this manner, a user is capable of inputting to the system throughout the keyboard 224, trackball 232 or mouse 226 and receiving output from the system via speaker 228 and display 238.

Preferred implementations of the invention include implementations as a computer system programmed to execute the method or methods described herein, and as a computer program product. According to the computer system implementation, sets of instructions for executing the method or methods are resident in the random access memory 214 of one or more computer systems configured generally as described above. Until required by the computer system, the set of instructions may be stored as a computer program product in another computer memory, for example in disk drive 220 (which may include a removable memory such as an optical disk or floppy disk for eventual use in the disk drive 220). Further, the computer program product can also be stored at another computer and transmitted when desired to the user's work station by a network or by an external network such as the Internet. One skilled in the art would appreciate that the physical storage of the sets of instructions physically changes the medium upon which it is stored so that the medium carries computer readable information. The changes in the medium may be physical, such as the changes in an optical disk, floppy disk, or hard disk, or they may be electrical, such as the changes in a magnetic tape or a magnetic disk.



or some other physical change. While it is convenient to describe the invention in terms of instructions, symbols, characters, or the like, the reader should remember that all of these and similar terms should be associated with the appropriate physical elements.

5 Note that the invention may describe terms such as comparing, validating, selecting, identifying, or other terms that could be associated with a human operator. However, for at least a number of the operations described herein which form part of at least one of the embodiments, no action by a human operator is desirable. The operations described are, in large part, machine operations processing electrical signals to generate other electrical signals.

10 Refer now to FIGURE 3 illustrating an HTTP server methodology 300 in accordance with the principles of the present invention. Methodology 300 may be incorporated, for example on server 110 in FIGURE 1. In step 302, HTTP server methodology 300 waits for a connection request on a TCP/IP socket. This is the idle state for HTTP server methodology 300. An artisan of ordinary skill in the art would understand a socket to be a virtual port that enables clients, such as clients 112-115, FIGURE 1, to connect to a server, such as server 110. In an embodiment of the present invention, the  
15 Internet Protocol (IP) address and port number for the socket are configurable. A client communicates information to server methodology 300 by specifying the preselected values of the IP address and the port number. On receipt of a connection request 304, a new socket is generated, step 306. Alternatively, a pool of sockets may be created, and a socket retrieved from the available pool. In step 308, a connection is established on the new socket created, or alternatively, retrieved, in  
20 step 306. In step 310, subprocess 301b waits for an HTTP request. Additionally, a timer is running whenever methodology 300 is waiting for an HTTP request. If the timer expires, step 311, the connection is deemed to have been broken, and the socket opened in step 308, is closed, step 313.

25 Upon arrival of an HTTP request, 312, it is then determined if the request is from an authorized user, step 314. A security server process for performing step 314 will be described in detail in conjunction with FIGURE 4.

30 If, in step 314, the user is valid, HTTP requests 312 is passed to a page server, step 316. The process by which a page server handles the HTTP request will be described in detail in conjunction with FIGURE 5. Note that a plurality of page servers may be running concurrently, for example, as multiple threads within methodology 300. In this way, methodology 300 may handle multiple requests. The appropriate page server may be determined in an embodiment of the present invention by the data passed in a Uniform Resource Locator (URL). URLs are a string representation of a resource available via the Internet. The syntax and semantics for URLs are defined in accordance with an Internet protocol standard in Request for Comments (RFC) 1738, which is hereby incorporated herein  
35 by reference. A portion of the URL is referred to as the "url-path". The url-path supplies the details of how a resource can be accessed. The syntax of the url-path is a sequence of alphanumeric strings separated by the "/" ("forward slash").

In an embodiment of the present invention, the url-path of HTTP request 312 is used to determine the page server that will be invoked upon receiving the request, and specify the functions being requested and the object upon which the functions will be performed. A url-path, in accordance

The "dir" values serve a similar function as directories do in a computer's local file system. The first directory, "dir<sub>1</sub>", is used to determine which page server will be invoked in step 316 to service the HTTP request. In other words, the particular value of "dir<sub>1</sub>" received in HTTP request 312 specifies the particular page server that will service the request. Page servers register themselves with methodology 300 during initialization. Initialization typically occurs when the program starts, usually a result of a static configuration read from a file or database. However, initialization may also occur, for example, in response to adding a new module to the equipment wherein the corresponding page server is configured and initialized dynamically. On registering, the page server provides the name of the directory the particular page server will "serve". Additionally, a brief description is also provided that is used to generate a "home page" for the tool corresponding to the particular page server. The "home page" allows users to access the tool management facilities using a conventional web browser. This will be discussed further in conjunction with FIGURE 5 hereinbelow.

The page server invoked in step 316 generates a reply to HTTP request 312. This will also be discussed further in conjunction with FIGURE 5. The reply is passed to methodology 300 which sends the reply in step 318. The reply may take a finite amount of time to send due to delays arising from available network resources, or transmission errors, for example. In step 321 methodology 300 waits for the send to complete. On completion, methodology 300 closes the socket opened/retrieved in step 306 and notifies client that send is complete.

Refer now to FIGURE 4 illustrating a security server process which may be used to implement step 314 in methodology 300. In step 402 it is determined if a cookie is received in the header of the HTTP request. If not, a login page is presented in step 404. If, however, a cookie is received in the HTTP header, in step 406 the user name is extracted and compared with a list maintained by the server.

In step 408 it is determined if the user is known. If a cookie has been received, and the user name extracted in step 406 compares with one of the names in the list, or if the user name and password received via the login page presented in step 404 compare with an entry in the list maintained by the server, then step 408 proceeds by the yes branch to step 410. In step 410, if a cookie was not received in step 402, that is the user logon via the login page presented in step 404, the user's browser is requested to set a cookie of the same name as the tool in the HTTP request, in which the cookie contains the user's name. Subsequent HTTP request will then contain the cookie in the header, and in servicing that request, methodology 300, FIGURE 3, via step 314 would proceed by the yes branch in the corresponding step 402. If, in step 408 the user was not recognized, then access fails, and in step 320 FIGURE 3, a request denied message is generated, and sent in step 318.

In step 412 it is determined if a check activity option is active. The check activity option automatically logs out a user that has been inactive for a preselected interval of time. If, in step 412, activity checking is enabled, then, in step 414 it is determined if the user has had recent activity. That is, if the user has been active prior to the expiration of the preselected time interval for automatically logging out users. If the user has not been recently active, then access fails, and, in step 320 FIGURE 3, a request denied message is generated, and sent in step 318.



If, however, in step 414 the user has generated activity within the preselected time interval, the contact time maintained in the server database is updated, step 416 and the current HTTP request is passed to the page servers, step 316, FIGURE 3. Furthermore, if, in step 412, activity checking is disabled, step 414 is bypassed, and methodology 300 implementing authentication checking in accordance with the embodiment of step 314 in FIGURE 4 proceeds directly to step 416 to update the contact time in the server database entry corresponding to the current user.

Refer now to FIGURE 5 illustrating a page server methodology 500 in accordance with the embodiment of the present invention. Page server methodology 500 may receive an HTTP request from step 316 of methodology 300, FIGURE 3.

In step 502 the HTTP request is inspected to determine the function to be performed. Because it is difficult to maintain accurate information about a client browser's session with an HTTP server, the functions that page server process 500 perform may only depend on the information passed in the HTTP request. The fields available in the HTTP request to store this information are, principally, are the URL, and the search (or query) field. In an embodiment of the present invention in accordance with page server process 500, FIGURE 5, the URL is used to describe the functions the page server is to perform.

As previously described in conjunction with FIGURE 3, the url-path in the url field in an HTTP request has the approximate form /dir<sub>1</sub>/dir<sub>2</sub>/dir<sub>3</sub>/... The field labeled "dir<sub>1</sub>" has a predetermined name which, appearing in a url-path informs the HTTP server process to invoke a page server process, in accordance with, for example, step 316 in FIGURE 3. In an embodiment of the present invention, the string "Explorer" may be used to invoke the page server process, however, a artisan of ordinary skill in the art would understand that another predetermined term in the "dir<sub>1</sub>" field of the url-path could be used, and such alternative embodiments would be within the spirit and scope of the present invention.

The next field, "dir<sub>2</sub>," may contain the "name" or similar designator of the function that is being requested. Thus, the function may be determined, in such an embodiment, in step 502 by inspecting the field "dir<sub>2</sub>" in the url-path of the url in the HTTP request. Additionally, the field "dir<sub>3</sub>" may contain the name or other identifier of the object upon which the function will be performed. An object identifier may simply be an assigned number which might, in an embodiment of the present invention start at one, or other predetermined value, and consecutively number the objects. Alternatively, the "fully qualified" object name (analogous to a fully qualified path name) may be used to build a hash table with the key being the hash of the object name and the value a pointer to the object, and the key serves as the object identifier. It would be understood by an artisan of ordinary skill that such alternative embodiments are within the spirit and scope of the present invention. The remaining fields in the url-path portion of the URL in the HTTP request may include additional function-specific information. These will be described further below.

Refer now to FIGURE 5-1 illustrating a tool object model 550 which may be used with the present invention. Object model 550 is in accordance with a tool object promulgated by Brook Automation, Inc. However, it would be understood by an artisan of ordinary skill in the tool automation art that the principles of the present invention may be used with other tool object models in the same fashion. Tool object model 550 provides a logical description of the automatic controller, or component of

a physical tool "on the factory floor." As indicated in Inset A, showing the generic format for describing the organization of tool object model 550, a property 552 is a collection of one or more objects 554; the "filled circle" denoting "collection." Thus, tool object model 550 includes a collection of one or more tool objects 556. Likewise, tool object 556 includes dictionaries 558 and resources 560.

5 Dictionary 558 is a collection of dictionary objects 562 which, itself, includes a collection of data definition objects 564. A data definition object defines a data structure, and may be viewed as a type statement analogous to a "typedef" in the C programming language. Resources 560 include a collection of one or more resource objects 566. Resource object 566 includes a collection of service objects 568.

10 Each service object 568 includes attributes 570, methods 572 and events 574. Attributes, which are defined as a collection of attribute objects 576 that a collection of one or more attribute objects 576. Methods 572 include a plurality of method objects 578. Method objects 578 perform the functionality in the management of the physical tool. Method objects 578 include error objects 580 for error reporting, and, in the typical fashion, inputs 582 and 584, each of which is defined to be a  
15 collection of one or more data item objects 586. A data item object 586 may be an actual value of a variable or parameter. Events 574 constitute a collection of one or more event objects 588, each of which includes an output which is a collection of one or more data item objects 586. Examples of attributes, methods and events will be described hereinbelow.

The functions that the page server may perform include an information function, an editing  
20 function, an execution function, and an expanding function. (These may be denoted "INFO", "EDIT", "EXECUTE" and "EXPAND" ) Each of these will be discussed below.

The INFO function provides detailed information about the current attributes of a selected object. The INFO function is invoked, step 504, in response to a corresponding first predetermined value in field "dir," representing the designator of the function being invoked. For example, the first  
25 predetermined value may be the string "Info" but other values would be used, as would be recognized by an artisan of ordinary skill. In response, INFO pages 506 containing the information about the attributes of the object (identified in field "dir<sub>2</sub>" of the URL) are returned to the client browser, step 508. Page server methodology then exits, step 510 by returning to step 321 of HTTP server process 300. At step 321, HTTP server process 300 waits for the send to complete, and then returns to step 302.  
30 for the next connect request.

This may be understood by referring to FIGURE 6.1 illustrating an exemplary display 602 for Generic tool tool object 604. Display 602 has been expanded to show child dependencies from root objects, as will be discussed in detail hereinbelow in conjunction with FIGURES 8.1 and 8.2. The dictionary object 606 includes Generic tool Dictionary 608. Within the Generic tool Dictionary 608  
35 there are a plurality of data definition objects 610 including the SEMI equipment communication standard (SECS) 612. SECS 612 itself includes a plurality of data items 614. Event reporting data item 616 includes a set of collection event items 618 and a set of report variables 620.

Information about a tool object may be obtained by sending a corresponding HTTP request to HTTP server process 300. FIGURE 3. HTTP server process 300 then passes the request to the  
40 page server process as previously described. Other aspects of the present invention will be apparent to those skilled in the art.

all of the details of every tool object in a large tool object model to a browser, and too error-prone to try to synchronize dynamic attributes of the object model between a client and the server.

Information about an object is obtained by sending an HTTP request to HTTP server methodology 300. FIGURE 3. An information HTTP request has a first predetermined value in the "dir," field of the URL-path. For example, the first predetermined value may be the string "Info". However, it would be understood by an artisan of ordinary skill that other predetermined values may be used in accordance with the principles of the present invention, and such alternative embodiments would be within the spirit and scope of the present invention. HTTP server methodology 300 passes the request to page server process 500, as previously described hereinabove. Referring again to FIGURE 5, inspect function step 502 uses the value in the "dir," field of the URL-path of HTTP request 312, corresponding to the information request, to pass the request to information step 504. Step 504 uses the value in the "dir," the object ID to generate the corresponding information page 506 and send the page to the client in step 508. An information HTTP request may be initiated by selecting the object for which the detailed information is sought by selecting the object on the web browser display. Thus, in exemplary display 602, chamber location process variables item 622, shown highlighted in FIGURE 6.1, has been selected. An object may be selected by "clicking" on the object on the display with a user input device, such as mouse 226, FIGURE 2.

In response to the information HTTP request, as previously described, page server process 500 generates the corresponding web page containing the detailed information for the selected object. FIGURE 6.2 illustrates exemplary information display 624 corresponding to chamber location process variables item 622, FIGURE 6.1. Exemplary information display 624 is with reference to the SECS, however, an ordinarily skilled artisan would understand that the principle of the present invention do not rely on the SECS, and tool objects implemented in accordance with another set of protocols may return different information. Nevertheless, the principles of page server process 500 would operate in the same fashion, and such alternative embodiments would be within the spirit and scope of the present invention.

Refer now to FIGURES 7.1 and 7.2 illustrating an exemplary display which may be shown on a client GUI in association with an edit attribute page 573, FIGURE 5, generated by page server process 500. Display 702 shows a tool management resource directory 704 including tool object 706. Tool object 706 provides a plurality of services that may be accessed using the principles of the present invention. Attributes of the services provided by the tool via the tool object model are also shown in display 702. Attributes 710 of GEM clock 708 are displayed, and include a time formatting attribute 712, shown highlighted in FIGURE 7.1. ("GEM" denotes the Generic Model for Communications and Control of SEMI Equipment standards promulgated by Semiconductor Equipment and Materials International (SEMI). SEMI is a global trade association that represents the semiconductor and flat panel which promulgates standards for ensuring compatibility equipment automation standards such as GEM. Exemplary embodiments will be described in the context of automated fabrication ("fab") facilities in the semiconductor industry. Nevertheless, it would be understood by an ordinary skilled artisan that the principles of the present invention may be used in

Attributes of objects may be modified using the principles of the present invention. An attribute may be modified by the client sending an EDIT request in an HTTP request 312. FIGURE 3. As previously described, page server process 500 inspects HTTP request 312 in step 502. If the HTTP request includes an edit request, determined in step 502 by inspecting the "dir<sub>1</sub>" field, for a second predetermined value, then in step 512, an edit method is invoked and an edit attribute page 512. is sent. step 508 The second predetermined value may be the string "Edit," however, an artisan of ordinary skill would understand that other values may be used. Additionally, the "dir<sub>2</sub>" field may include the name of the variable being edited.

FIGURE 7.2 illustrates a graphical display 703 of an edit response in accordance with the principles of the present invention. Display 703 corresponds to the exemplary sixteen character time format attribute 712 of FIGURE 7.1. Description field 714 describes the particular attribute. Name field 716 displays the name of the attribute, which reflects the attribute name appearing in the corresponding display 702. Read-Only field 718 displays the value of a Boolean "Read-Only" value. If the value displayed in Read-Only field 718, then the value of the attribute may not be changed by the user, conversely, if the value displayed in Read-Only field 718 is "False", then the value of the attribute itself may be modified by the user as in the exemplary attribute in FIGURES 7.1 and 7.2. Value field 720 displays the value of the attribute itself. In FIGURE 7.2, the value of the attribute, "False" is contained in an editable form 724. The user may, in the exemplary attribute of FIGURES 7.1 and 7.2, change the value of the attribute, and thereby change the format for time values. VarType field 722 displays the attribute type, in the exemplary illustration of FIGURE 7.2, a Visual Basic Boolean value. The user may change the value of an attribute that is not "Read-Only" by entering the new value in form 724, as shown, and returning a message to the HTTP server process 300, FIGURE 3, by invoking the eval and save control 726. The user may invoke control 726 by "clicking" on control 726 using, for example, mouse 226, FIGURE 2. Restore value control 728 allows the user clear user entered values from form 724, and restore values sent down with the form, if any

An edit request may be an HTTP request 312 in which the value of "dir<sub>1</sub>" has the value "Edit" or other third predetermined value in forming page server process 500 that the EDIT service is to be performed. As previously described, field "dir<sub>2</sub>" contains an object identifier, in this instance, the identifier of the object whose attribute is to be modified. The value in the "dir<sub>2</sub>" field of the uri-path of HTTP request 312 may identify the attribute field to be modified. In the exemplary attribute of FIGURE 7.2, the field which may be modified is the attribute value field 712, as previously described. Additionally, the new value of the attribute may be included in a query portion of the HTTP request. Thus an exemplary URL may be `http://server.port/Explorer/edit/objectID/FieldName` with query string `fieldName = value0 & fieldName = value1`. In response, the server sets the value of the elements of variable "fieldName" (in this example a two element array) to value0 and value1. Query strings formatted in this way specify named value pairs. However, alternative embodiments may be implemented in which the requested function and object ID are embedded in the query string, and additionally, may be formatted as an XML query. It would be recognized that such alternative embodiments would be within the spirit and scope of the present invention. Page server

methodology 500 then sends a new edit attribute page 512 with the updated value to the client via another step 508.

Because a tool object model that may be used with the present invention may be highly configurable, page server methodology 500 accommodates a variety of models configured for various tools and service combinations. In responding to an information request, as discussed above in conjunction with step 504, the response sent in step 508 passes to the client a few root objects. Other tool objects in the tool object model are accessed by exploring parent-child relationships from the root objects or the descendants thereof. Child information may be provided by page server methodology 500 in response to an HTTP request 312 having a fourth predetermined value for "dir<sub>1</sub>". In an embodiment of the present invention, the fourth predetermined value of "dir<sub>1</sub>" may be the string "Extend" however other predetermined values may be used, as an artisan of ordinary skill would understand. If HTTP request 312 includes the fourth predetermined value for "dir<sub>1</sub>", inspection step 502 passes the request to expand step 514 which generates and EXTEND model response 516. EXTEND model response 516 contains only the child object names and the relation to the parent object, identified by the object identifier in field "dir<sub>2</sub>" in the corresponding HTTP request 312. Additional details about a child object may be obtained by invoking information step 504 through an INFO request as described hereinabove.

Thus, for example, consider FIGURE 8.1 illustrating a graphical display operable for initiating an EXTEND request. Display 802 illustrates a plurality of attributes associated with Generic tool tool 706 described hereinabove in conjunction with FIGURES 7.1 and 7.2. Highlighted in display 802 is SECSStandard data definition object 804. (SECSStandard data definition objects refer to the SEMI Equipment Communications Standard (SECS) promulgated by the SEMI. SECS is described further hereinbelow in conjunction with FIGURES 9.1 and 9.2.) If a user were interested in the subdefinitions of the SECS Standard data definition object 804, an embodiment of the present invention, "clicking" on triangle 806 would initiate an HTTP request, such as HTTP request 312, FIGURE 3, including the third predetermined value in field "dir<sub>1</sub>". Because triangle 806 is associated with SECSStandard data definition object 804, the value in field "dir<sub>2</sub>" in the HTTP request may be the string "SECSStandard". Although the exemplary display 802 and 808, FIGURE 8.2, below, are shown with respect to a SECS implementation of tool automation, an artisan would understand that the principles of the present invention illustrated thereby may be used in alternative embodiments employing other automation protocol. It would also be understood that such alternative embodiments would be within the spirit and scope of the present invention.

In response, inspect function step 502 passes the corresponding HTTP request 312 to step 514 and EXTEND model page 516 generates display 808 illustrating in FIGURE 8.2 on the users display. Beneath SECSStandard data definition object 804 appears a plurality of subdefinitions 810.

Additionally, execution of tool object methods may be initiated via page server methodology 500. Refer now to FIGURE 9.1 illustrating a portion of an INFO display 902. A directory of methods 904 includes start method 906 (shown highlighted) in FIGURE 9.1. In an embodiment of the present invention, start method 906 may be a method of the SECS Initiate Processing Service

35 358. INFO display 902 may be generated by page server methodology 500 in response to an HTTP



hereinabove in conjunction with FIGURE 6.1. In response, page server methodology 500 passes the request to INFO step 504 which generates an INFO page 506 which produces display 910 in FIGURE 9.2. Display 910 includes information about a method, for example, description field 912 and service name field 914, which, in the exemplary display 910, indicates the SECS initiate processing service 908.

An artisan of ordinary skill would recognize that the principles of the present invention do not rely on the particular detailed information shown in exemplary INFO display 910, and it would be further recognized that alternative tool object models may include additional information in the corresponding display, or omit some of the detailed information included in exemplary display 910.

Additionally, display 910 includes execute method control 916. In the embodiment of FIGURES 9.1 and 9.2, execute method control 916 may be used to initiate processing on a tool. For example, a user by "clicking" on execute method control 916 may initiate an HTTP request which is received by page server methodology 500. An EXECUTE HTTP request generated in response to user input via execute method control 916, may include a fourth predetermined value in field "dir<sub>1</sub>". In an embodiment of the present invention, the fourth predetermined value may be the string "Execute"; however, it would be understood by an artisan of ordinary skill that other predetermined values may be used, and such alternative embodiments would be within the spirit and the scope of the present invention.

In response to the fourth predetermined value in the "dir<sub>1</sub>" field in the HTTP request, inspection step 502 passes the request to execute step 518. In step 518, page server process 500 invokes the method 520 corresponding to the objectID received in the EXECUTE request whereby a message is sent to the tool corresponding to the tool object in the HTTP request, such as HTTP request 312. The method implementation itself may be supplied by the tool manufacturer or other third party supplier. The tool object may be specified by an object identifier (ID) value in "dir<sub>2</sub>" field in the URL-path portion of the HTTP request. In an embodiment of the present invention, page server methodology 500 may identify the tool corresponding to the objectID via a table maintained on the data processing system executing page server methodology 500.

Method execution may be further understood by referring to FIGURE 5.2 illustrating remote processing control method 521, which may be one of the plurality of methods 520 in FIGURE 5. In step 523, an initiate processing request is sent to the tool. In step 525, methodology 521 waits for an initiate processing acknowledge from the tool. On receipt of the reply, methodology 521, in step 527, awaits for an event report from the tool. The creation of an event report, and the display thereof is discussed hereinbelow in conjunction with FIGURES 10.1 and 10.2. In step 529, following receipt of the event report, sends a message to the tool acknowledging the receipt of the event report. If in step 531, processing is complete, methodology 521 exits. Otherwise, the method loops over steps 527 and 529, until processing completes.

Messages may be sent to the tool using a standard communications protocol, for example the Transmission Control Protocol/Internet Protocol (TCP/IP) via a local area network, such as LAN 124, FIGURE 1. The message format in such an embodiment may be specified by the

High Speed SECS Message Specification (HSMC) developed by the Tool HSMC Consortium.



Semiconductor Equipment and Materials International (SEMI). In such an embodiment, the format of the message sent by the method, for example method 521, FIGURE 5.2 may be in accordance with SECS encoding in which the message is specified in a two-byte integer format. The first byte denotes a category, or "stream", and the second byte defines the function to be performed by the tool. For example, the initiate processing request, sent in step 523, FIGURE 5.2, may be denoted "S2F27", where the "S" indicates that the first value corresponds to the stream, and the second value corresponds to the function.

In an alternative embodiment, the server implementing page server process 500, such as server 101 in FIGURE 1, may communicate with the tool using a point-to-point message transfer protocol, such as SECS-I. SECS-I is also promulgated by SEMI. In such an embodiment, server 101 may be connected to each of tools 111 via an RS-232 serial port. (An artisan of ordinary skill would understand that an RS-232 compatible serial port implements signal levels in accordance with the RS-232 standard promulgated by the Electronics Industries Association (EIA).) It would be recognized by an artisan of ordinary skill that the principles of the present invention do not rely on the particular communications protocol used to send messages to the tool, and, a protocol in accordance with the HSMS specification or, alternatively, SEC-I would be within the spirit and scope of the present invention.

Although the EXECUTE function has been described in the context of initiating processing, particular tool management protocol, such as the GEM standard, may provide a plurality of control and monitoring methods. An artisan of ordinary skill would recognize that these may also be invoked using an execute request with the corresponding object ID in the "dir<sub>2</sub>" field of the URL.

Additionally, a method, such as one or more of methods 520, FIGURE 5, receives messages from the tools, such as tools 111 in FIGURE 1. For example, a "Create Report" method which may be invoked by sending the corresponding HTTP request, may define a "report" (a selected set of variables whose values are to be reported), link the report to a particular event on the tool and enable that event whereby a message is sent by the tool on the occurrence of the specified event. As described hereinabove, such messages may be received by remote processing method 521, and cached by the server performing the method, such as server 110, FIGURE 1. Now, recall that in the exemplary display 808 discussed in conjunction with FIGURE 8.2, a plurality of child objects 810 is shown. The plurality of child objects 810 includes an event reporting object 812. Referring now to FIGURE 10.1, there is illustrated therein a message history display portion 1020. Display portion 1020 is a child object of event reporting object 812, FIGURE 8.2 and may be displayed by executing the expand function, step 514 in page server methodology 500, FIGURE 5 as previously described. Display portion 1020 shows a plurality of message histories 1022. In the exemplary embodiment of FIGURE 10.1 in which messages are sent in accordance with the SECS-II format, the messages are indicated using the "SmFn" nomenclature. Message history 1024 corresponds to the initiate processing request discussed in conjunction with FIGURES 9.1 and 9.2. Associated with message history 1024 are a plurality of child objects 1026. These may be displayed as illustrated in FIGURE 10.1 by executing the EXPAND step 514 of page server methodology 500 by initiating a corresponding HTTP request, as described hereinabove in conjunction with FIGURES 9.1 and 9.2. An artisan of ordinary skill would recognize that the principles of the present invention do not rely on the particular format of the messages sent between the server and the tool.

"PID" daughter object 1028 and the "MID" daughter object 1030 correspond to values in the respective fields shown in message information display 910, fields 916 and 918 respectively

Furthermore, detailed information about messages (messages are objects) may be obtained by initiating an information HTTP request, as discussed hereinabove in conjunction with FIGURES 6.1, 6.2, as well as 9.1 and 9.2. The corresponding HTTP request includes the second predetermined value, for example, "Info" in the "dir." field of the url-path portion of the HTTP request, and the object identifier of the selected message, for example, one of the corresponding message history, for example one of message history 1022, FIGURE 10.1, in the "dir<sub>2</sub>" field of the request. In response, page server methodology 500 passes the request to step 504, FIGURE 5, and a corresponding information page 506 is sent in step 508. FIGURE 10.2 illustrates information display 1028 corresponding to message history 1024 in an embodiment of the present invention in which messages conform to the SECS-II specification. However, it would be understood by an artisan of ordinary skill in the art that the principles of the present invention embodied in page server methodology 500 for displaying message information do not rely on the SECS definitions, and other predetermined, message formats may be used in similar fashion. It also would be understood by persons of ordinary skill in the art that embodiments of the present invention using other such predetermined message formats would be within the spirit and scope of the present invention.

The present invention also allows for user customization of methods associated with the tool objects. Returning to FIGURE 5, the method invoked by execute step 518, in response to a corresponding HTTP request, may be a default method provided, for example, by the particular tool objection implementation used. Alternatively, the present invention provides a mechanism by which the default mechanism typically compiled code, may be overridden. In FIGURE 11, a flowchart of override process 1100 is shown. Override process 1100 is performed for each object in the tool object model. Each object in the tool object model includes a *script override* attribute. This attribute contains script source code. (If the default method is to be used for a particular object, the length of the script source may be zero.) In step 1110, for the corresponding object, an override registry list is opened. In an embodiment of the present invention, the list may contain a Boolean variable for each method associated with the object in which the Boolean value may be initialized to "FALSE", indicating that the method has not been overridden.

If override process 1100 is being performed on an initialization, or if an override attribute of an object has been modified by the user, step 1112, the script source is parsed in step 1114. In an embodiment of the present invention, the scripting language may be VBScript (Visual Basic Scripting Edition, a scripting language developed by Microsoft Corporation). However, other scripting languages could be used, and would be within the spirit and scope of the present invention. In step 1116, it is determined if the script contains a method having a matching signature as a default method associated with the object. In an object oriented program paradigm, such as that used in the present invention, each method has a name, a return type (which specifies a type of a value returned by the method), and, optimally, a set of arguments. Taken together, these form the "signature" of the method. If in step 1116, a matching signature is found, then the Boolean value "TRUE" is stored in the object's override registry list opened in step 1110. In an embodiment of the present invention, the

method. If, however, in step 1116, no matching signature is found, then the initial Boolean value of "FALSE" remains in the registry list entry, and signals that the default method has not been overridden. Override process 1110 then returns to step 1112 to determine if the override attribute is subsequently modified. Process 1100 loops until the override attribute is modified, or a reinitialization has occurred.

In this way, a user can customize methods that act on the tool objects without recompiling code. Additionally, the script source code does not need to be parsed each time a method is invoked. Returning to FIGURE 5, when execute step 318 invokes a method, it refers to the override registry list to determine if the method has been overridden. If not, it invokes the, typically compiled, default method. Otherwise, it executes the override script.

Although the present invention and its advantages have been described in detail, it should be understood that various changes, substitutions and alterations can be made herein without departing from the spirit and scope of the invention as defined by the appended claims.

## CLAIMS:

1. A tool management method comprising the steps of:  
receiving a first request via a network;  
determining a type of said first request using a first predetermined field in a portion of said  
5 request; and  
sending a first message to a tool in response to said request and said request type wherein  
said first message is operable for controlling an action of the tool.

2. The method of claim 1 further comprising the step of determining an identification of a tool  
10 object corresponding to said tool using a second predetermined field in said portion of said request.

3. The method of claim 1 wherein said tool performs a first action in response to said message.

4. The method of claim 3 wherein said tool returns at least one second message associated with  
15 said first action, said method further comprising the step of caching said at least one second  
message.

5. The method of claim 4 further comprising the steps of:  
retrieving selected ones of said at least one second message; and  
20 generating a response to a second request using said selected ones of said at least one  
second message.

6. The method of claim 5 further comprising the step of sending said response to a client system  
initiating said first and second requests.  
25

7. The method of claim 1 wherein said first request is transferred in accordance with a network  
transfer protocol, and said network transfer protocol defining a format of said first request.

8. The method of claim 1 further comprising the steps of  
30 receiving a connection request, and  
opening a connection to a client, said connection being operable for communicating requests  
and responses to said requests

9. The method of claim 1 further comprising the steps of:  
35 receiving a second request, said second request selected from the group consisting of  
information (INFO) requests, expand requests and edit requests, wherein, in response to each of said  
information requests, a set of selected data for a tool object corresponding to a managed tool is  
loaded into a WorldWideWeb ("Web") page for sending to a client, in response to each of said edit  
requests, a Web page having portion operable for user entry of one or more values for modifying a  
40 tool object attribute is generated for sending to said client, and in response to each of said expand

requests a set of child object names and relations to a parent object identified in each expand request is generated for sending to said client.

10. The method of claim 1 wherein said type of said first request denotes an execute request.

11. The method of claim 1 wherein said step of sending said first message is in response to execution of a tool object method identified in said first request.

12. The method of claim 11 further comprising the step of overriding said tool object method.

13. The method of claim 12 wherein said step of overriding said tool object method comprises the steps of:

parsing a script source;

determining if said script source includes a method signature matching a method signature of said tool object method; and

if so, executing a corresponding portion of said script source

14. The method of claim 7 wherein said transfer protocol is the hypertext transfer protocol (HTTP), and said first portion corresponds to a first field in a uniform resource locator (URL) path.

15. A data processing system comprising:

circuitry operable for receiving a first request via a network;

circuitry operable for determining a type of said first request using a first predetermined field in a portion of said request; and

circuitry operable for sending a first message to a tool in response to said first request and said type of said first request wherein said first message is operable for controlling an action of the tool.

16. The data processing system of claim 15 further comprising the circuitry for determining an identification of a tool object corresponding to said tool using a second predetermined field in said portion of said request.

17. The data processing system of claim 15 wherein said tool performs a first action in response to said message

18. The data processing system of claim 17 wherein said tool returns at least one second message associated with said first action, said method further comprising the step of caching said at least one second message.

19. The data processing system of claim 18 further comprising:

circuitry operable for retrieving selected ones of said at least one second message; and  
circuitry operable for generating a response to a second request using said selected ones of  
said at least one second message.

5        20.        The data processing system of claim 19 further comprising circuitry operable for sending said  
response to a client system initiating said first and second requests.

21.        The data processing system of claim 15 further comprising:  
circuitry operable for receiving a connection request; and  
10        circuitry operable for opening a connection to a client, said connection being operable for  
communicating requests and responses to said requests.

22.        The data processing system of claim 15 further comprising circuitry operable for receiving a  
second request, said second request selected from the group consisting of information (INFO)  
15        requests, expand requests and edit requests, wherein, in response to each of said information  
requests, a set of selected data for a tool object corresponding to a managed tool is loaded into a  
WorldWideWeb ("Web") page for sending to a client, in response to each of said edit requests, a Web  
page having portion operable for user entry of one or more values for modifying a tool object attribute  
is generated for sending to said client, and in response to each of said expand requests a set of child  
20        object names and relations to a parent object identified in each expand request is generated for  
sending to said client.

23.        The data processing system of claim 15 wherein said type of said first request denotes an  
execute request  
25

24.        The data processing system of claim 15 wherein said step of sending said first message is in  
response to execution of a tool object method identified in said first request.

25.        The data processing system of claim 24 further comprising circuitry operable for overriding  
30        said tool object method.

26        The data processing system of claim 25 wherein said circuitry operable for overriding said tool  
object method comprises  
circuitry operable for parsing a script source.  
35        circuitry operable for determining if said script source includes a method signature matching a  
method signature of said tool object method, and  
circuitry operable for executing a corresponding portion of said script source, if so.

27        A computer program product embodied in a tangible storage medium, the program product  
40        embodied as a program of instructions for performing the steps of



receiving a first request via a network;

determining a type of said first request using a first predetermined field in a portion of said request; and

5 sending a first message to a tool in response to said first request and said type of said first request, wherein said first message is operable for controlling an action of the tool.

10 28. The program product of claim 27 further comprising instructions for performing the step of determining an identification of a tool object corresponding to said tool using a second predetermined field in said portion of said request.

29. The program product of claim 27 wherein said tool performs a first action in response to said message.

15 30. The program product of claim 29 wherein said tool returns at least one second message associated with said first action, said method further comprising the step of caching said at least one second message

20 31. The program product of claim 30 further comprising instructions for performing the steps of: retrieving selected ones of said at least one second message; and generating a response to a second request using said selected ones of said at least one second message

25 32. The program product of claim 31 further comprising instructions for performing the step of sending said response to a client system initiating said first and second requests.

30 33. The program product of claim 27 further comprising instructions for performing the steps of receiving a connection request; and opening a connection to a client, said connection being operable for communicating requests and responses to said requests.

35 34. The program product of claim 27 further comprising instructions for performing the step of receiving a second request, said second request selected from the group consisting of information (INFO) requests, expand requests and edit requests, wherein, in response to each of said information requests, a set of selected data for a tool object corresponding to a managed tool is loaded into a WorldWideWeb ("Web") page for sending to a client, in response to each of said edit requests, a Web page having portion operable for user entry of one or more values for modifying a tool object attribute is generated for sending to said client, and in response to each of said expand requests a set of child object names and relations to a parent object identified in each expand request is generated for sending to said client

35. The program product of claim 27 wherein said type of said first request denotes an execute request.

5 36. The program product of claim 35 wherein said instructions for performing the step of sending said first message are performed in response to execution of a tool object method identified in said first request.

10 37. The program product of claim 36 further comprising instructions for performing the step of overriding said tool object method.

38. The program product of claim 37 wherein said instructions for performing the step of overriding said tool object method comprises instructions for performing the steps of:  
parsing a script source;  
determining if said script source includes a method signature matching a method signature of  
15 said tool object method; and  
if so, executing a corresponding portion of said script source.

39. The method of claim 1 further comprising the step of:  
receiving a second request; and  
20 loading a set of selected data for a tool object corresponding to a managed tool into a WorldWideWeb ("Web") page for sending to a client in response to said second request.

40. The method of claim 39 wherein said Web page has a portion operable for user entry of one or more values for modifying a tool object attribute.

25 41. The data processing system of claim 15 further comprising  
circuitry operable for receiving a second request; and  
circuitry operable for loading a set of selected data for a tool object corresponding to a managed tool into a WorldWideWeb ("Web") page for sending to a client in response to said second  
30 request.

42. The data processing system of claim 43 wherein said Web page has a portion operable for user entry of one or more values for modifying a tool object attribute.

35 43. The program product of claim 27 further comprising instructions for performing the steps of  
receiving a second request; and  
loading a set of selected data for a tool object corresponding to a managed tool into a WorldWideWeb ("Web") page for sending to a client in response to said second request

44. The program product of claim 41 wherein said Web page has a portion operable for user entry of one or more values for modifying a tool object attribute.

1/15

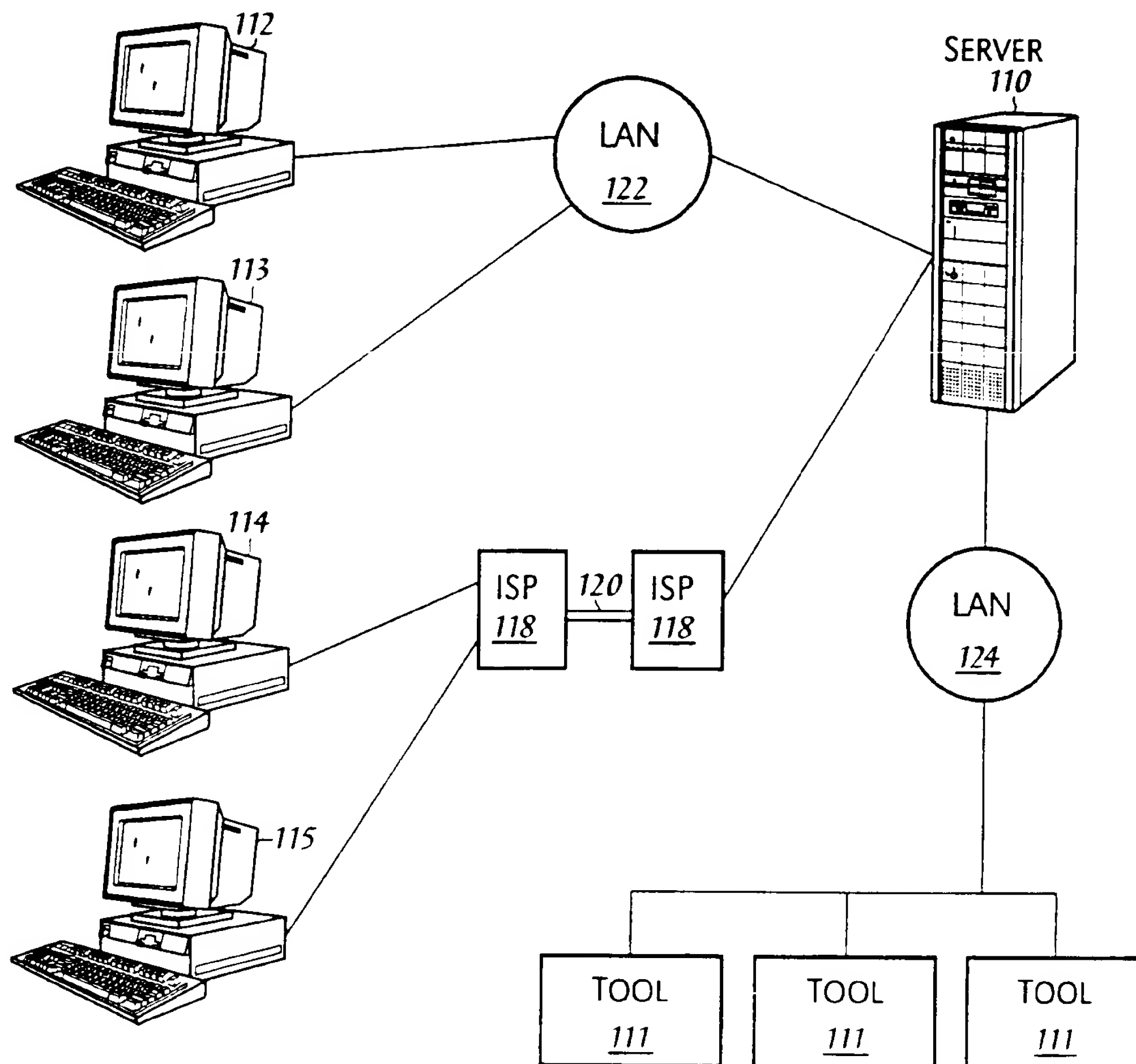


Fig. 1

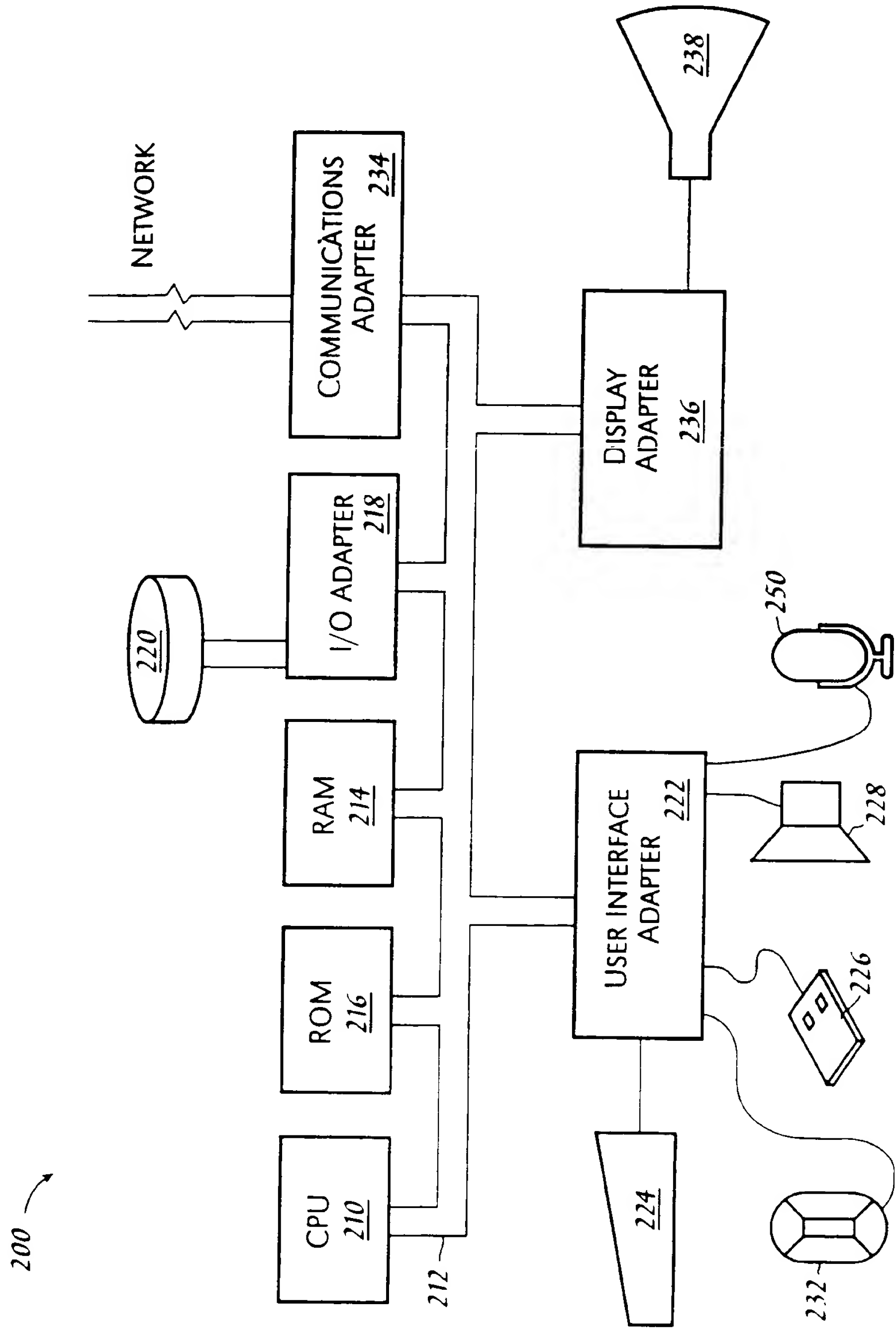


Fig. 2

3/15

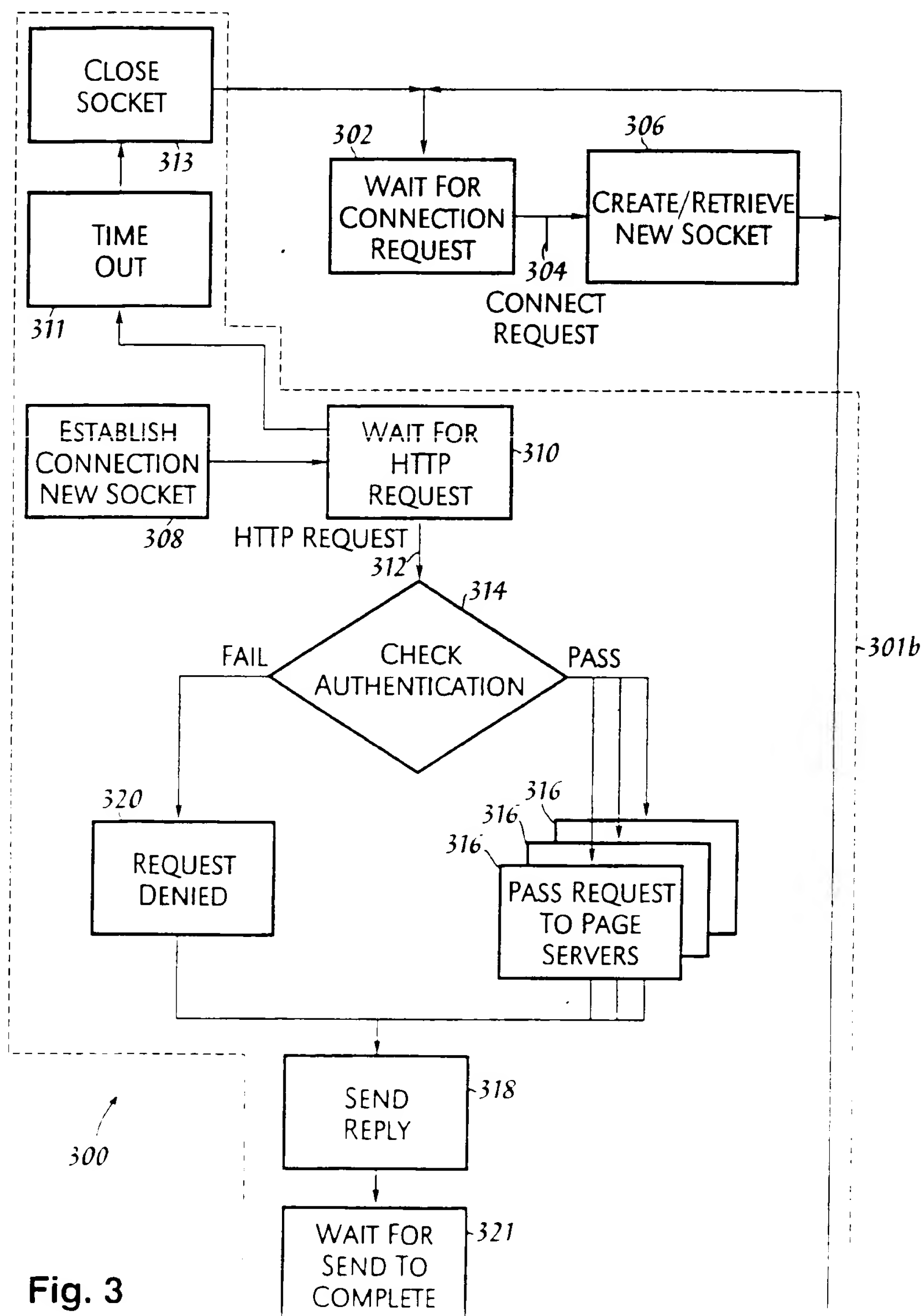


Fig. 3



4/15

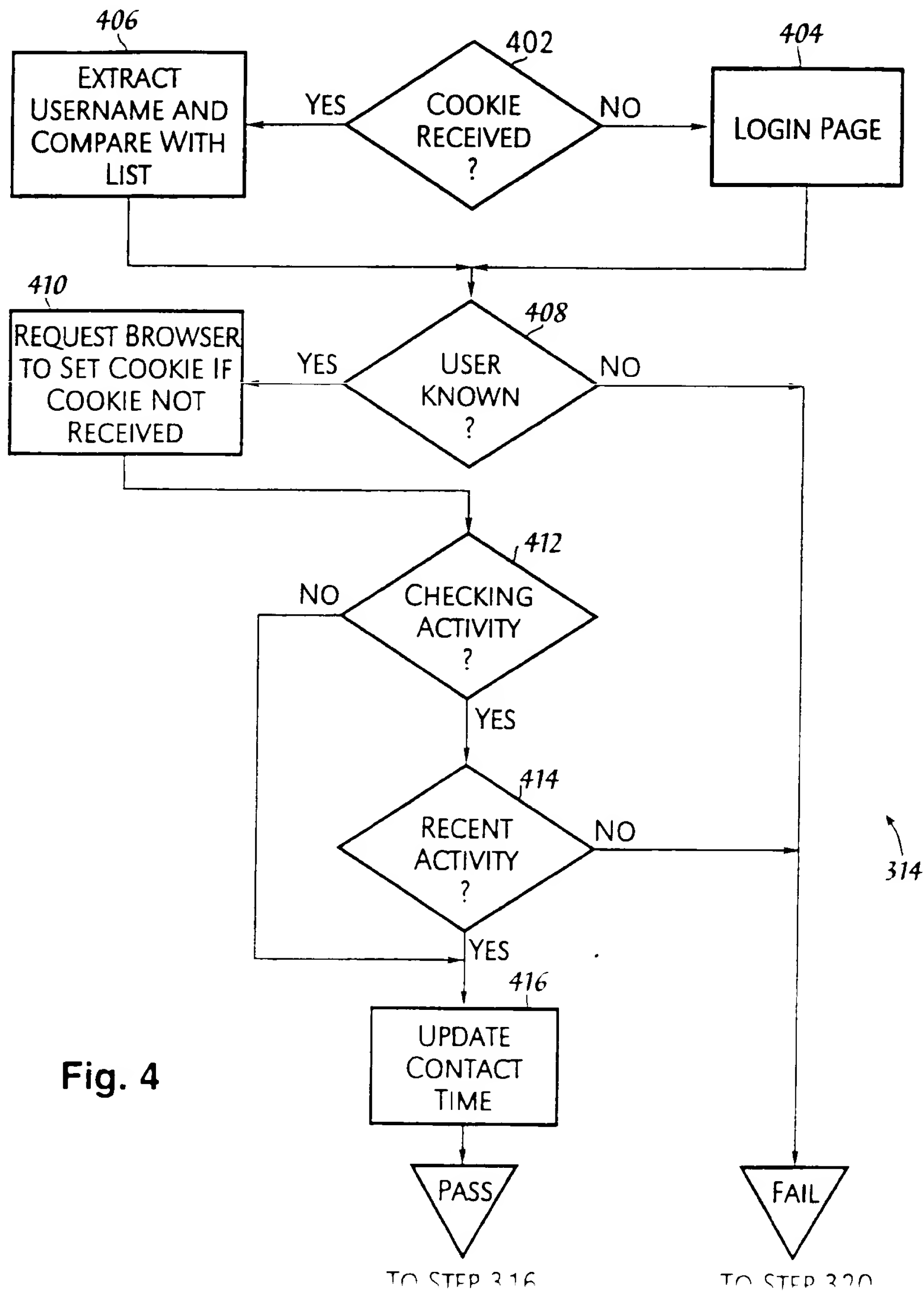


Fig. 4

5/15

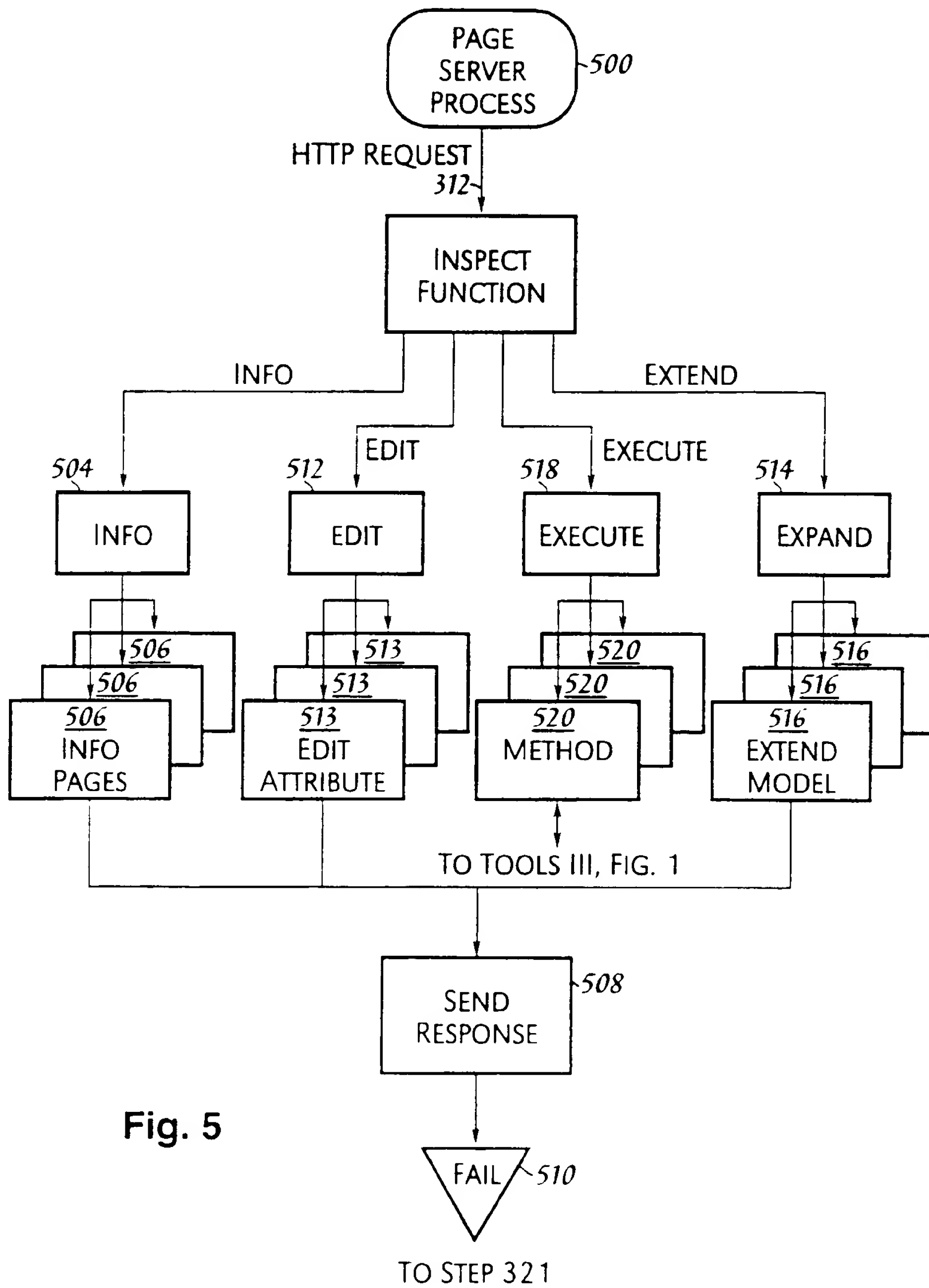
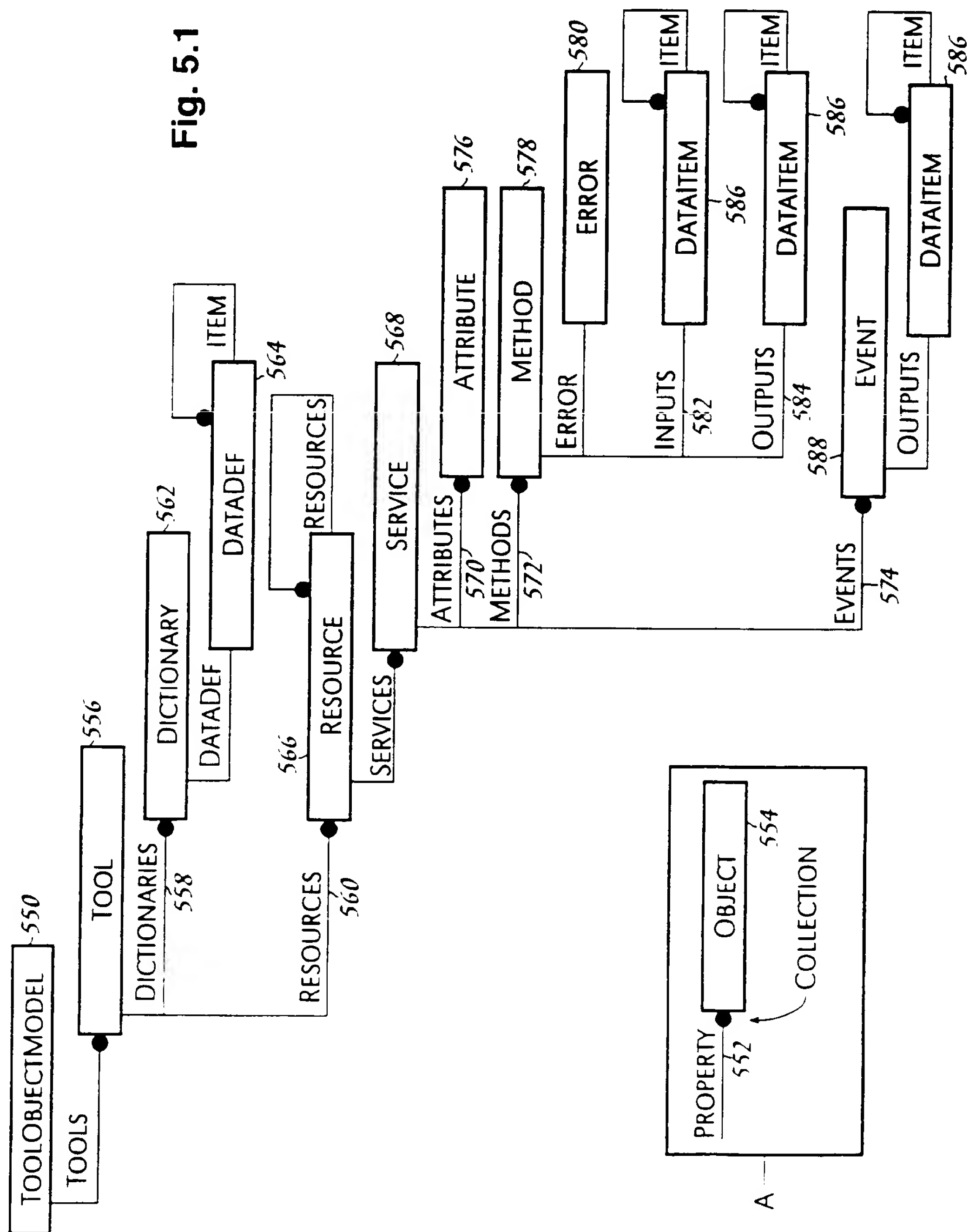


Fig. 5

**Fig. 5.1**



7/15

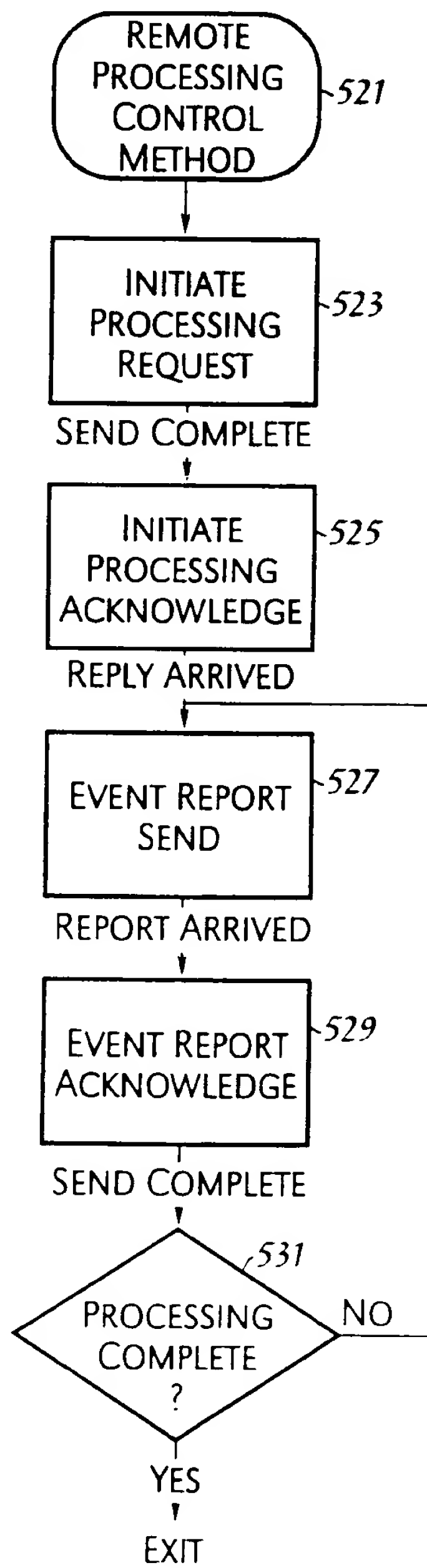
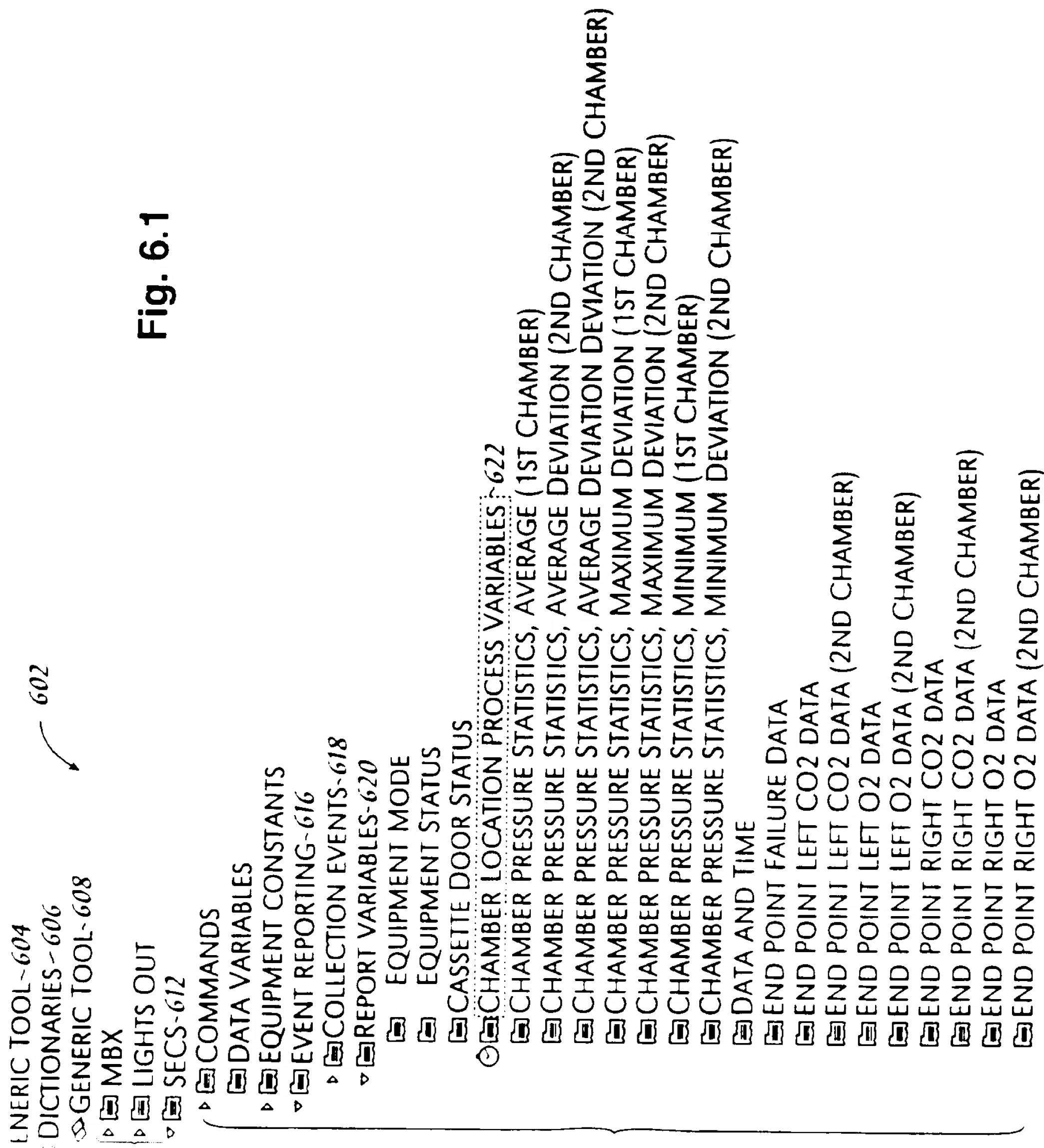
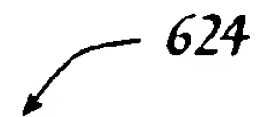


Fig. 5.2

Fig. 6.1



9/15

624

Alias ("Intrinsic")	CHAMBER LOCATION PROCESS VARIABLES
AccessID	49
AllowDuplicates	FALSE
CanAddName	FALSE
Comments	0 = BACK CHAMBER INSTALLED; 1 = RIGHT CHAMBER INSTALLED; 2 = RIGHT AND BACK CHAMBERS INSTALLED
Count	0
DefaultValue	(NULL)
Description	
Dictionary Name	GENERIC TOOL
Format	26 (TOMFORMAT12)
Key	157
LinkChildren	FALSE
Maximum	(NULL)
Minimum	(NULL)
Name	CHAMBER LOCATION PROCESS VARIABLES
OrderChildrenByAccessID	FALSE
Parent Name	REPORT VARIABLES
SimulatedValue	(NULL)
Units	
ValueRequired	FALSE

Fig. 6.2



10/15

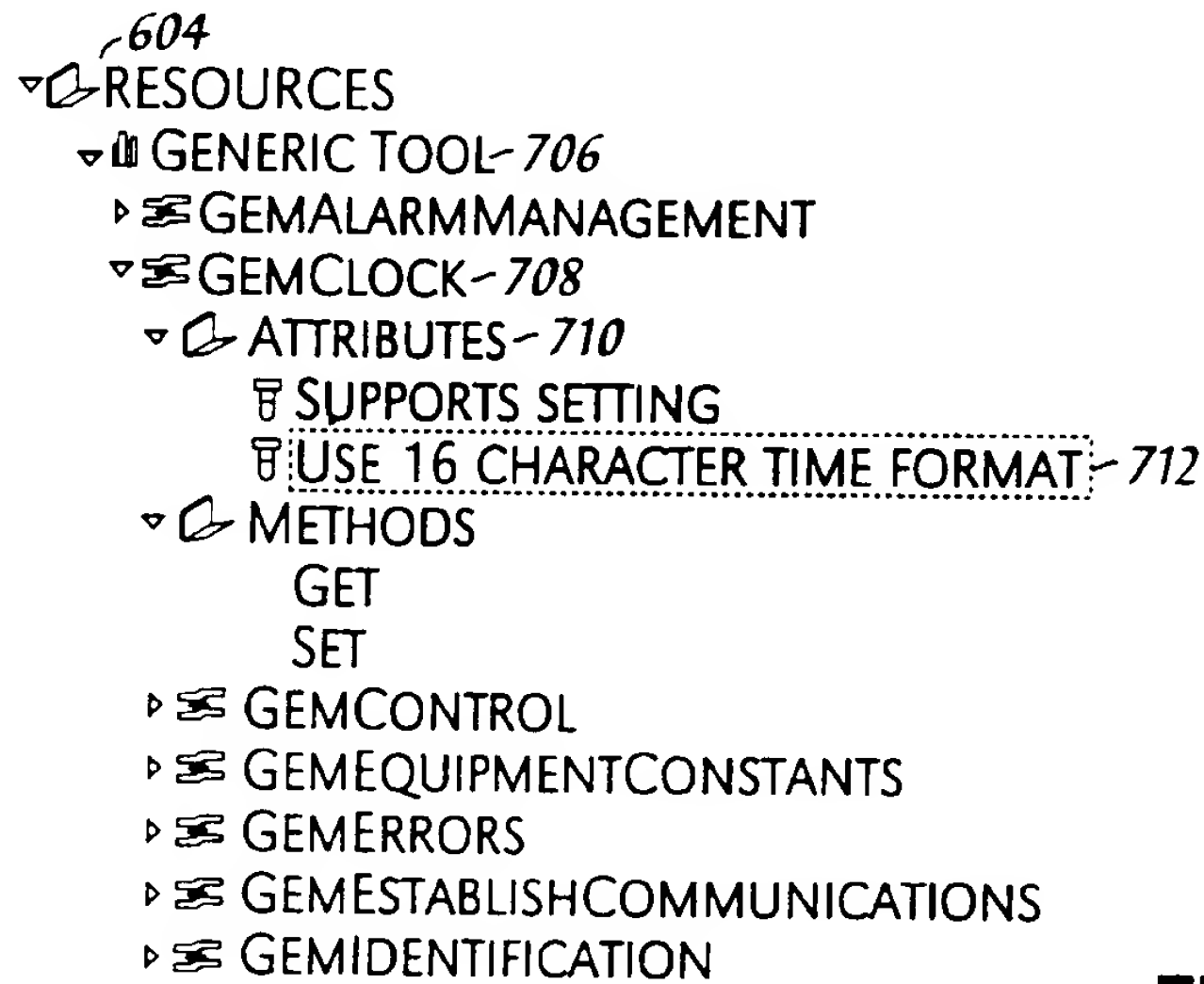


Fig. 7.1



Fig. 7.2

DESCRIPTION 714 IF TRUE, USE THE YYYYMMDDhhmmsscc FORMAT FOR TIME VALUES OVER THE SECS INTERFACE. IF FALSE, USE THE YYMMDDhhmmss FORMAT

NAME 716 USE 16 CHARACTER TIME FORMAT

READONLY 718 FALSE

VALUE

724

FALSE

EVAL AND SAVE

RESTORE VALUE

VARTYPE-722

11 (vbBoolean)

726

728

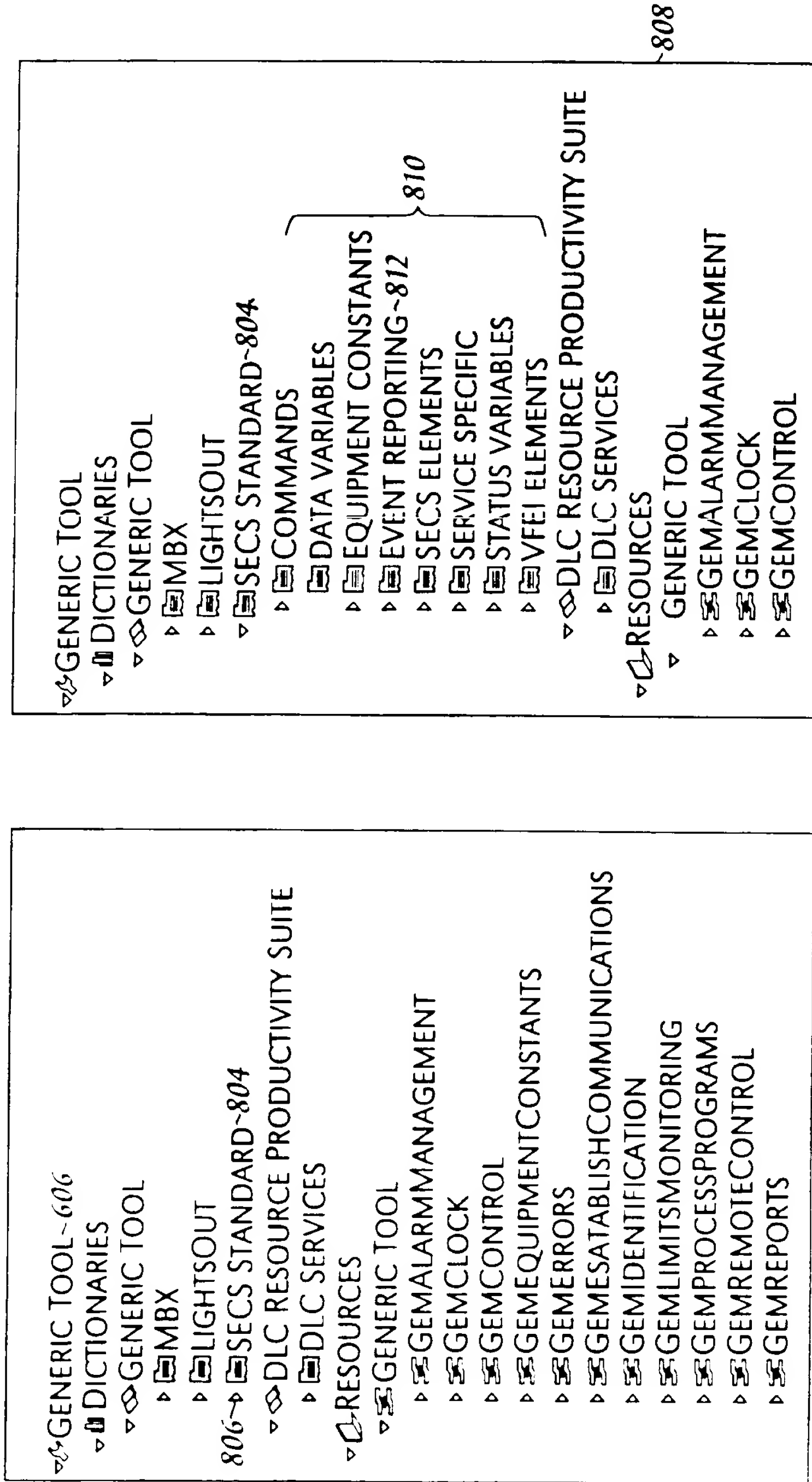
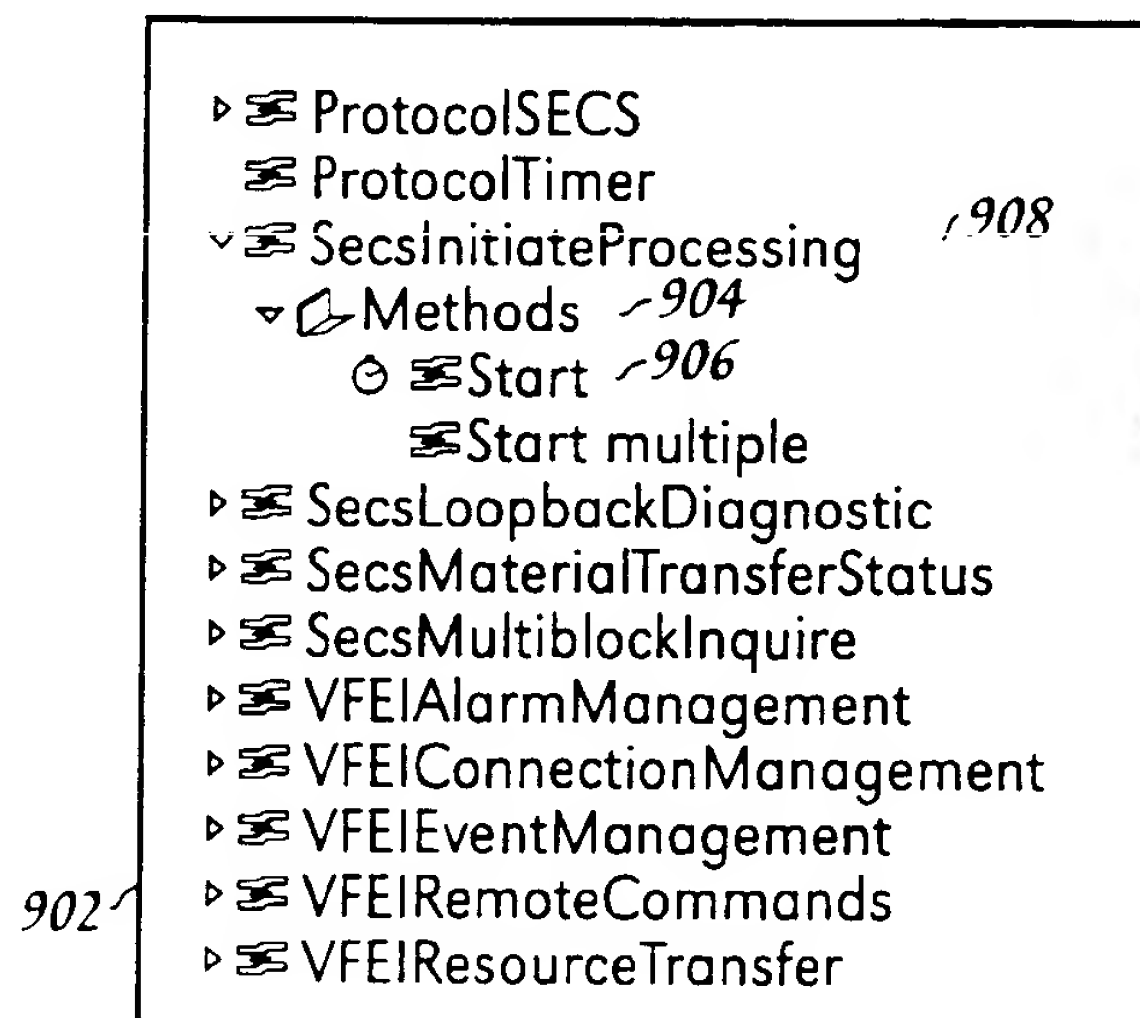


Fig. 8.1

Fig. 8.2

12/15

**Fig. 9.1**

Inputs

LOC1

PPIDM28\_03\_1

MIDS437186.2

910

Outputs

(NONE)

Method info

CLONED

DATECOMPLETED

DATESTARTED

DESCRIPTION

ENABLED

INPROGRESS

NAME

NOTIFY

SERVICE.NAME

SUPPRESSENABLECHECK

TAG

FALSE

0 (NEVER)

0 (NEVER)

START PROCESSING

TRUE

FALSE

START

2 (tomNotifyAlways)

SecsinitiateProcessing

FALSE

MAKE CHANGE

RESTORE VALUE

EVAL AND SAVE

RESTORE VALUE

VERIFIED

FALSE

EXECUTE METHOD

916

Fig. 9.2

14/15

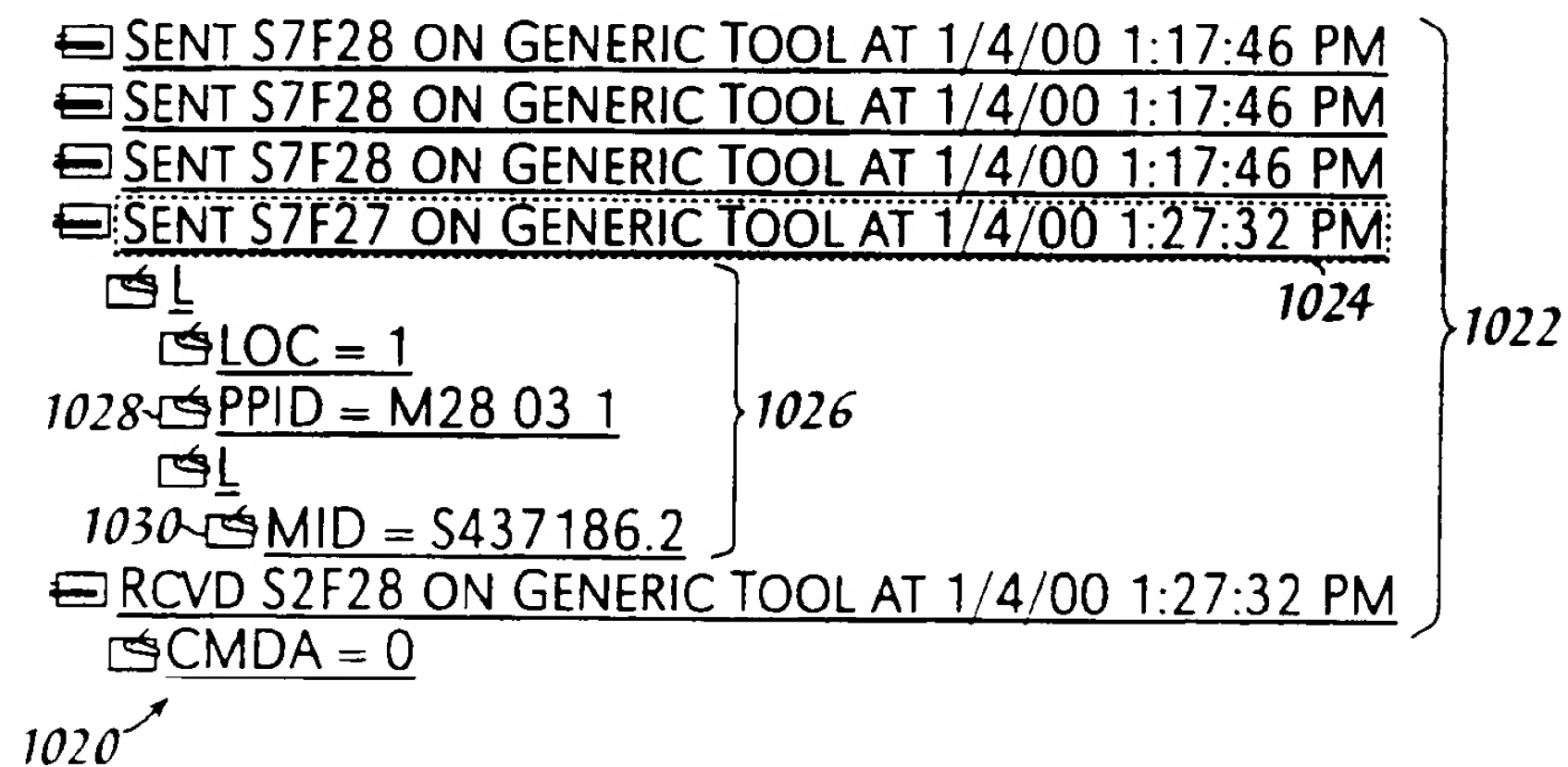


Fig. 10.1

Description	Initiate Procesing Request
ResourceItem.Count	1
ResourceName	Generic Tool
Length	28
MessageType	SECS
Name	S2F27
Raw	01 03 21 01 01 41 08 4D 32 38 5F 30 33 5F 31 01 01 41 09 53 34 33 37 31 38 36 2E 32
ReplyExpected	True
Sent	True
TransactionID	25

Fig. 10.2

15/15

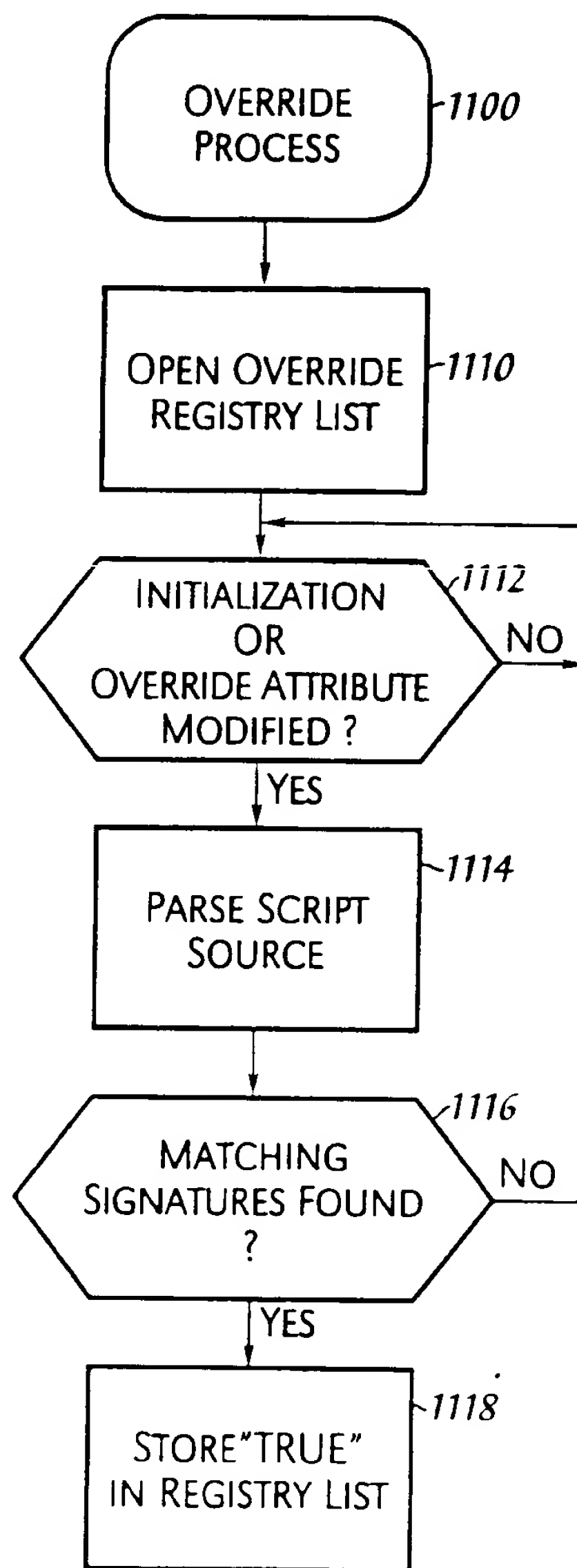


Fig. 11